Midterm Notebook

Welcome to your midterm. Logistics:

- Open book/open note/**no internet**
- You are not allowed to discuss the exam with each other
- All questions about the exam will come to me, through email. Do not send any public messages to me, or each other about the exam.
- If there are any clarifications required, I will post them on brightspace and update this document.

A note on the **kinds** of answers I expect: As is our style on HW and in class, many of these questions are open ended and are **not** asking you to repeat what you've read or heard in class. On the contrary, if I read my own words (or a texts) I will mark that down! I expect you to demonstrate your original thoughts. Almost none of these questions require 3-word answers (some do though, those should be clear by the question!). Having said that, I also don't want you to start just typing out vocabulary words that we've used in class.

🤔 **Tip**: If you feel you can't answer a question, **skip it and come back**. Sometimes reading the entire thing will help clarify the individual parts. If all else fails, I will award partial credit for effort, and a clear explanation of what you're confused about and why.

🤔 **Tip**: If you're really confused, **try and thoroughly explain your confusion!**

Changelog

If there are any updates/errors, I will update this section and email the class.

👾 **Note:**

- This is **version 2**, updated on 2022-10-10 at 6pm for use on Colab
  - Colab uses an old version of python (from 2019!) and an old version of `sklearn` so I had to update the last problem
- ~~This is **version 1**, updated on 2022-10-10 at 9:00am.~~

Notebook Setup

In [2]:
```
# imports↔
```
Out[2]:

Problem 0

Lets do a simple exercise using `numpy` !

0.A

Write some code to specify an array representing a point in "the **unit-cube** in 3-D space. That is, give me a single **random** point whose (x, y, z) values are each between (0,1).

In [3]:
```
a_point = # EDIT HERE
a_point
```
Out[3]:
```
array([0.04425985, 0.32989572, 0.17696902])
```

0.B

Now, give me a single array with `1000` many points in the unit cube:

In [4]:
```
random_points = # EDIT HERE
random_points.shape
```
Out[4]:
```
(1000, 3)
```

0.C

Give me the **minimum**, **maximum** and **mean** of **each column**, to verify that we've done this correctly:

In [5]:
```
# EDIT HERE

min_vals, max_vals, mean_vals
```
Out[5]:
```
(array([0.00170532, 0.00135462, 0.0008504 ]),
 array([0.99933413, 0.99779327, 0.99876069]),
 array([0.49478026, 0.49629412, 0.5132123 ]))
```

0.D

🤔 **Pause-and-Ponder:** According to the above, where is the **center** of this cube? Does this make sense? Why or why not? Comment below!

Type *Markdown* and LaTeX: α^2

0.E

Great! Now lets **shift** the entire cube of points, so that its **new center**, is $(2, 1, 4)$

In [6]:
```
shifted_points = # EDIT HERE
```

0.F

As we did above, lets verify that the new center of this cube is in the correct place!

In [7]:
```
shifted_mean_vals = # EDIT HERE
shifted_mean_vals
```

Out[7]:
```
array([1.97912106, 0.99258823, 4.10569839])
```

0.G

And finally, **how far on average**, according to the *usual* distance measure (which we fancily called the $\mathcal{L}_2$-norm) is the original single point, from the new shifted cube?

🙋🏽‍♀️ **Hint** I'm asking you to calculate the $\mathcal{L}_2$ norm between the original point, and **each point in the shifted cube**, and then average all those distances together.

In [8]:
```
# EDIT HERE
```

0.H

🤔 **Pause-and-Ponder:** Does this make sense given how you shifted the points in the first place? Think about it!

Type *Markdown* and LaTeX: α^2

Problem 1

For this problem, we're going to use the `penguins` dataset built into `seaborn` .

In [9]:
```
penguins_df = sns.load_dataset("penguins").dropna()
```

Note, this gives you a dataframe as:

In [10]:
```
penguins_df
```

Out[10]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	s
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Ma
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Fema
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Fema
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Fema
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	Ma
...	...	...	...	...	...	...	
338	Gentoo	Biscoe	47.2	13.7	214.0	4925.0	Fema
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Fema
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Ma
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Fema
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Ma

333 rows × 7 columns

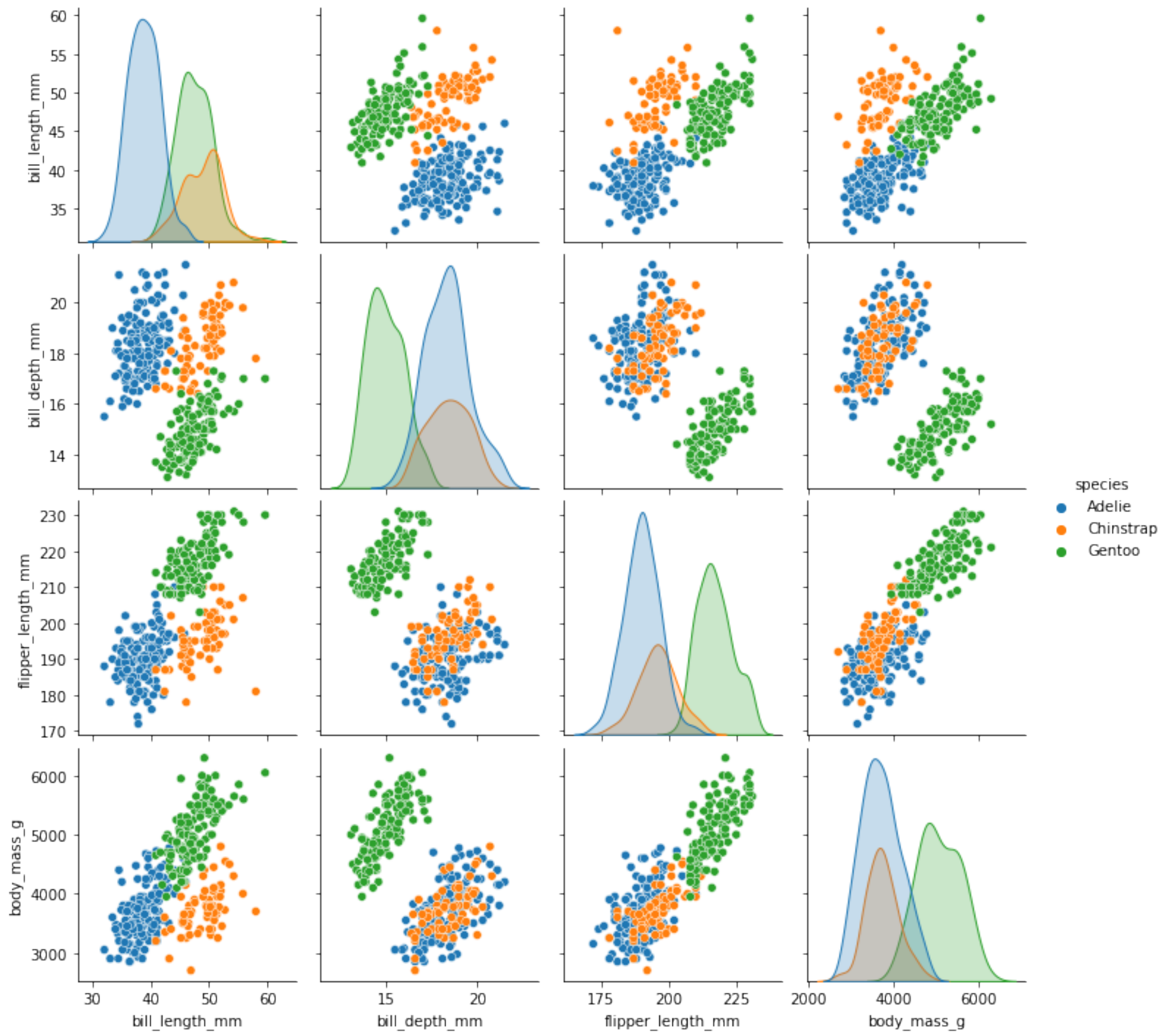
1.A

As we did in the "Exploratory Data Analysis" discussion, draw a "scatter plot matrix" of this dataset's 4 numeric features:

- `bill_length_mm`
- `bill_depth_mm`
- `flipper_length_mm`
- `body_mass_g`

and colored by their `species`.

👇 **Note:** This is the figure I want you to generate



In [11]:
```
# EDIT HERE
```

🤔 **Question:** Explain what we are looking at in this figure. Explain to me its diagonal, and its off diagonal elements. Why is this kind of graphic helpful? Give a thorough explanation here!

Type *Markdown* and LaTeX: α^2

🤔 **Question:** Based on the figure and your explanation, if I wanted to tell apart the **Adelie** from the **Chinstrap** penguis, **which two features would be the best?** Why? Give a good explanation here!

Type *Markdown* and LaTeX: α^2

1.B

Now that you've picked these two features, perform a **linear regression** and use it to try and classify these two penguins apart.

Make sure you report the weight values you've learned!

☝️ **Hint:** This is very similar to what we did on HW1!

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

1.C

Show a figure, plotting the datapoints with your regression line.

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

1.D

Perform a **logistic regression** and use it to try and classify these two penguins apart. **Make sure you report the value's you've learned!**

☝️ **Hint:** Remember, these are available under the `.coef_` and `.intercept_` names for logistic regression in `sklearn`

In [12]:
```
from sklearn.linear_model import LogisticRegression
```

In [ ]:

In [ ]:

In [ ]:

Bonus

Show a 2D figure, plotting the datapoints with your logistic regression line.

In [ ]:

In [ ]:

In [ ]:

1.E

Calculate the error in both the linear regression and logistic regression models. Which performs better?

In [ ]:

In [ ]:

In [ ]:

1.F

🤔 **Question:** Wait a minute. How did we use linear regression and logistic regression for **classification**? Explain why these are called regression models, and how we used them for classification!

Type *Markdown* and LaTeX: α^2

Problem 2

For this problem, we are going to do a basic **exploration** of a KNN, along the lines of how we did it in class.

I will give you the code to create/train one. I just want you to think conceptually like we did in class about KNNs.

To make this part simpler, I will assume above you already generated a dataframe containing only **Adelie** and **Chinstrap** penguins.

Copy/paste it below:

In [13]:
```
penguins_df = # EDIT HERE
penguins_df
```

Out[13]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	M
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Fen
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Fen
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Fen
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	M
...	...	...	...	...	...	...	
215	Chinstrap	Dream	55.8	19.8	207.0	4000.0	M
216	Chinstrap	Dream	43.5	18.1	202.0	3400.0	Fen
217	Chinstrap	Dream	49.6	18.2	193.0	3775.0	M
218	Chinstrap	Dream	50.8	19.0	210.0	4100.0	M
219	Chinstrap	Dream	50.2	18.7	198.0	3775.0	Fen

214 rows × 7 columns

Now, we can define a list of **feature names** we are interested in looking at:

In [14]:
```
feature_names = ['bill_length_mm', 'bill_depth_mm']
```

Now, we can use that to **select those features** from of our dataframe along with labels:

In [15]:
```
X = penguins_df[feature_names].values
y = penguins_df['species'].replace({'Adelie':0,'Chinstrap':1}).value
```

🤩 Now we're ready to train a KNN!

In [16]:
```
from sklearn.neighbors import KNeighborsClassifier
```

Lets set the very important `n_neighbors` parameter. For right now, lets jsut set it to `10` and we will experiment with it later:

In [17]:
```
n_neighbors = 10
```

Lets fit it!

In [18]:
```
knn_model = KNeighborsClassifier(n_neighbors)
knn_model.fit(X, y)
```

Out[18]:
```
▾        KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

Lets see how we did by calculating our **accuracy** on our entire dataset:

In [19]:
```
print(f'Accuracy: {knn_model.score(X, y)*100:.2f}%')
```

```
Accuracy: 95.79%
```

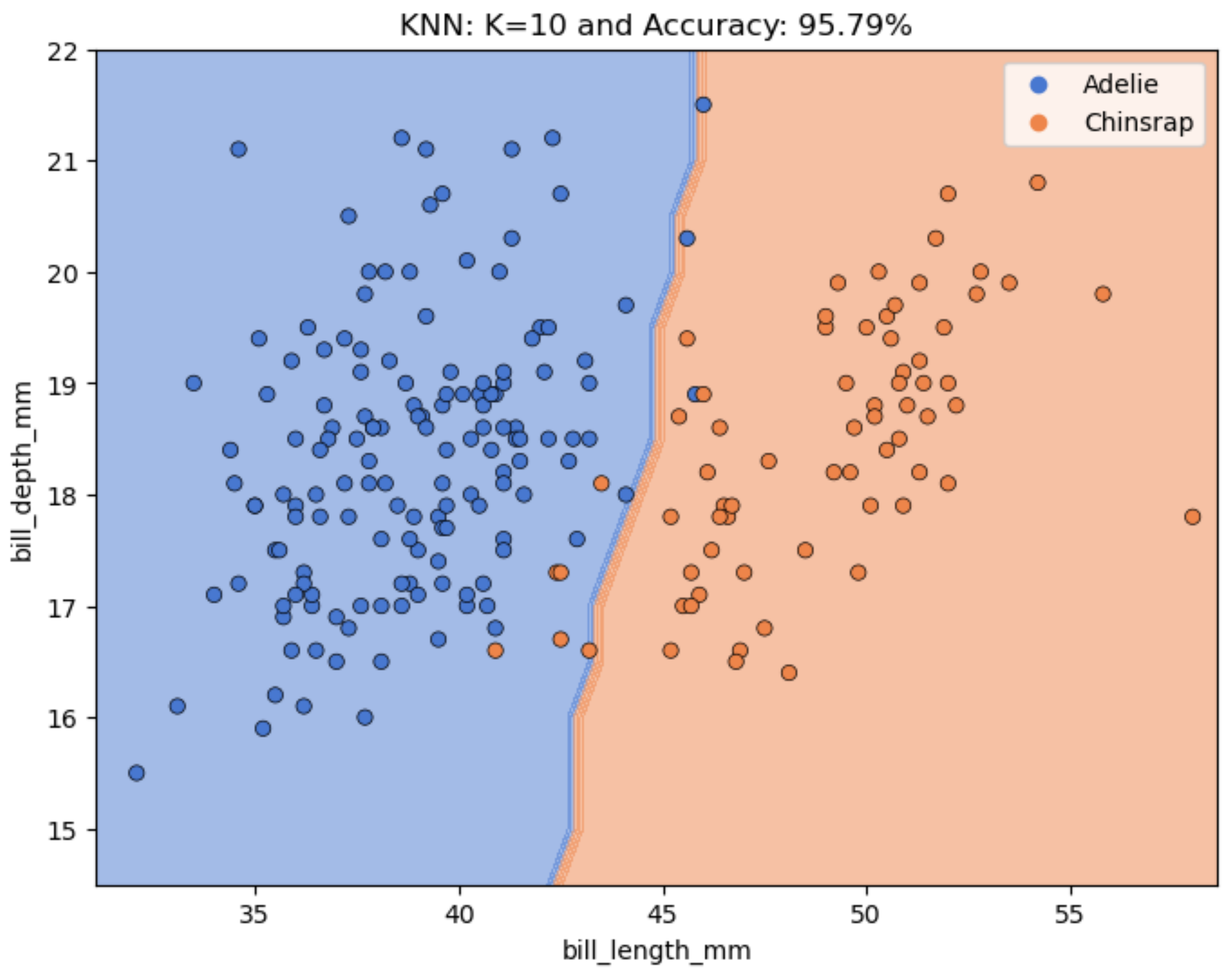
Hm. This seems pretty good, but lets visualize it!

☝🏽 **Note:** This is where I updated this NB to version 2 to account for an old Colab version.

In [20]:
```
# decision boundary plotting function↔
```

Now we can call it:

In [21]:
```
plot_decision_boundary(knn_model,X,y)
```



Now lets use it on two new features:

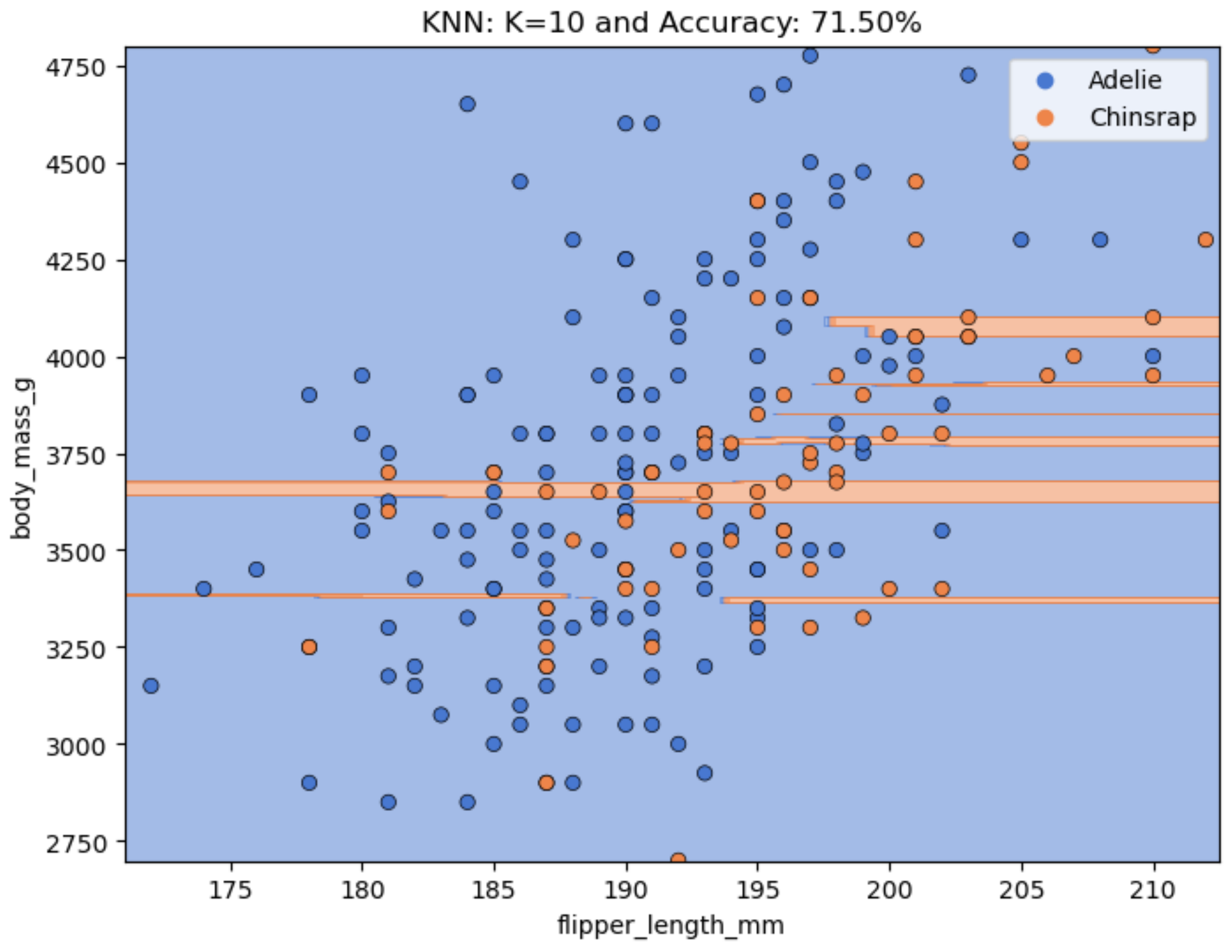
- `flipper_length_mm`
- `body_mass_g`

In [22]:
```
feature_names = ['flipper_length_mm', 'body_mass_g']
X, y = penguins_df[feature_names].values, y
n_neighbors = 10

knn_model = KNeighborsClassifier(n_neighbors)
knn_model.fit(X, y)
```

Out[22]:
```
▾         KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

In [23]:
```
plot_decision_boundary(knn_model,X,y)
```



2.A

🤔 **Pause-and-Ponder:** What are we looking at here? Is this good or bad? Why do you think so? Give a **thorough** explanation here!

Hint: Pay attention to the *scale* of our axes!

Type *Markdown* and LaTeX: α^2

2.B

Now, go back, and repeat this experiment with different groups of features and different values for **K**/ `n_neighbors` . Report the pair of features used, the `K` used, and what you observed for the accuracy:

Features Used	K	Accuracy
?	?	?
?	?	?
?	?	?
?	?	?
?	?	?
?	?	?

2.C

🤔 **Pause-and-Ponder:** Does this match waht you thought would happen from the exploratory section above? Why or why not? Comment below!

Type *Markdown* and LaTeX: α^2

Bonus Problems

Below are 2 bonus problems. You may attempt both, but **only serious attempts will be graded!** If you attempt them do so seriously! Short/simple answers will not receive any points!

Bonus - 1

For this problem, implement a simple gradient descent loop to solve the same **linear regression** that you solved above!

🙋🏽‍♀️ **Hint:** In class, we gave the **gradient** for Linear-Regression! Just use it as you normally would in gradient descent!

In [24]:
```
# EDIT HERE
```

In [25]:
```
# EDIT HERE
```

🤔 **Pause-and-Ponder:** Compare your answer to the one above. Should they match? Why or why not? Comment below!

Type *Markdown* and LaTeX: α^2

Bonus - 2

How does the material we've learned so far relate to any other material you've seen before? Any similarities? Differences?

If this is your first exposure to the material, then how does this relate to what you thought it was going to be before taking the class? How about what you wanted to use ML for in the future?

- Note: I'm **not** asking about style, difficulty, or coverage in the class!
- This question is **not a request for feedback**!

It's asking you to reconcile the material we've learned in class with the other material you've learned so far in other classes/your career, or what you thought it was going to be! I'm looking for things like:

- "In class A we learned about the Bias-Variance trade-off like this, which is different than how we learned about it here. They are different in this way, they are similar in this way. I'm having a hard time seeing how they are talking about the same thing because ... etc.
- I haven't had any previous exposure to these topics before. I didn't realize what a big deal it was to select an appropriate hypothesis space and algorithm, etc.
- I always heard deep learning just works. But how come they don't overfit all the time? etc.
- I'm confused about how exactly Linear regression can ever do better than KNNs, since KNNs use distance to .... etc.
- In my lab project we always use ___ method, but that doesn't seem to fit with what we are learning about, etc.

I want you to think deeply about this question. This exercise causes you to re-evaluate what you know, in terms of what you're learning and vice-versa, often changing your interpretation of both!

☝️ **Note: This bonus is "open ended", and I will award points based on the depth of your answer.**

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2
